

**University of California at Berkeley**  
**College of Engineering**  
**Department of Electrical Engineering and Computer Sciences**

EECS150  
Spring 2002

J. Wawrzynek  
E. Caspi

## **Project Checkpoint: Audio Output (DAC+Amp)**

### **1 Objectives**

The primary objectives of this lab are:

- Learn how to interface to a Digital-to-Analog Converter (DAC)
- Equip your project board with a complete audio output stage

The hardware side of this lab involves wire-wrapping a digital-to-analog converter (DAC) and audio amplifier to your Xilinx board, along with several discrete components (resistors, capacitors). The software side of this lab involves writing a serial interface to the DAC and a simple waveform generator with which to test the DAC interface.

### **2 Digital-to-Analog Converter (DAC)**

A DAC is a device that converts a sequence of numbers into an analog waveform. Our synthesizer needs a DAC to convert sound samples into an electric signal, which, with proper amplification, can drive a loudspeaker or headphones.

We will use the Analog Devices *AD1866* dual 16-bit audio DAC. A datasheet and wire-wrap ID guide will be available on the course web page. This particular DAC supports two audio channels, but we will use only the left channel (L). The DAC converts 16-bit, two's-complement words into output voltage levels. Each word must be shifted into the DAC serially (via pin DL, "Data Left"), MSB first, in-sync with a clock signal (CLK). DAC samples each input on the *rising-edge of its clock*. Once an entire word is shifted in, the falling edge of a control signal (LL, "Latch Left") instructs the DAC to convert that word into a voltage. We will send 16-bit samples to the DAC at a sample rate matching the MIDI baud rate, namely 31.25kHz. The bits of each sample can be serially shifted into the DAC at a rate of up to 8MHz.

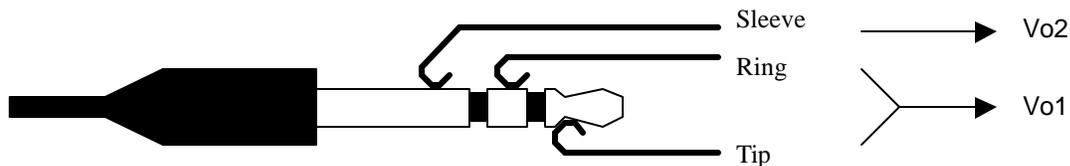
NOTE: The DAC shifts in its input DL on the *rising-edge of its clock*. The circuit shifts the data out to the DAC most significant bit first. If you use an ordinary shift register, your signals will appear a small delay after the rising edge of your clock. Therefore, to allow the DAC to correctly sample your data, the DAC's clock input should be the inverse of the circuit clock (thereby shifting the process of sampling to the middle of the cycle, where data is stable).

### 3 Audio Amplifier

The DAC's analog output is too weak to drive a loudspeaker or headphones on its own. It must be passed through an amplifier that can supply enough current and power to make audible sound.

We will use the National Semiconductor *LM4862 Boomer 675mW* audio power amplifier. A datasheet and wire-wrap ID guide will be available on the course web page. One interesting feature of this IC is its SHUTDOWN input can turn off the chip output. We will drive that pin from the FPGA to control when the amplifier turns on. Your design should drive the SHUTDOWN pin low to keep the amp on. The Xilinx I/O pins default to logic-high when the FPGA is powered-on, providing a mechanism to keep the amplifier off until it is configured with your design.

The amplifier's differential outputs ( $V_{o1}$ ,  $V_{o2}$ ) should be connected to the 1/8-inch phones connector on the Xilinx board. Our phones connector has 3 conductors (tip, ring, sleeve) to support stereo sound. The tip and ring, which represent the left and right audio channels, should be tied together using wire-wrap. Connect one amp output to the tip-ring pair and the other amp output to the sleeve.



1/8-inch phones connector + hookup to amplifier

### 4 Lab Assignment

For the hardware part of this lab, you must wire-wrap the DAC, amplifier, and discrete components to your project board. There are two discrete packs, one per IC. Try to place the ICs (DAC and amp) close to the periphery of the wire-wrap area, where they can be close to a power/ground pin pair (labeled “- +” on the board). Straddling each such pin pair is a  $0.1\mu\text{F}$  bypass capacitor that can provide a small burst of current to an IC's power/ground pins. This capacitor helps keep an IC's supply voltage constant as its power draw fluctuates. If an IC is not close to the board periphery, connect a bypass capacitor from one of the discrete packs between the IC's power and ground pins. Bypass capacitors should always be as close as possible to an IC (ideally, soldered on), since the inductance of wire connections diminishes their effectiveness.

The DAC and amplifier must be connected to the pins in the table below.

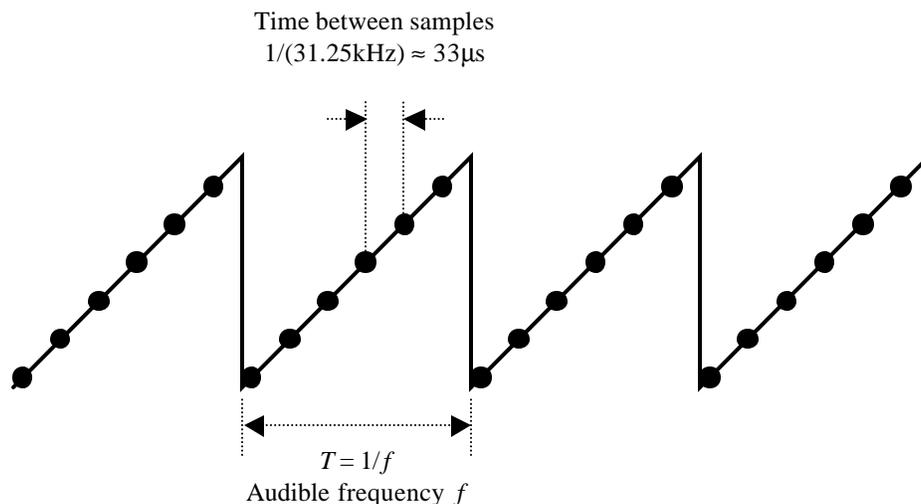
Signal	FPGA Pin
DAC LD	72
DAC LL	69
Amp SHUTDOWN	70
DAC CLK (8MHz)	41

**Pin connections for DAC and amplifier**

A wiring schematic for the DAC and amplifier is attached. All resistors and capacitors in the schematic are soldered to the discrete packs. They must be connected as shown in order for the ICs to function properly.

For the software part of this lab, you must construct two circuit designs:

1. A *parallel-to-serial converter* to drive the DAC's LD and LL inputs. This may be done in a variety of ways, typically using a shift register and some additional logic. Keep in mind that a new 16-bit sample must be fed to the DAC every 32μs so as to match the sample rate with the MIDI baud rate of 31.25kHz. Thus LL should send a down-edge once every 32μs, immediately following a 16-bit serial transmission to LD. As was noted previously, you may shift data to the DAC at a rate not exceeding 8MHz. Consult the DAC data sheet for precise timing information. Mistiming the LL down edge will result in failure to shift-in the sample's LSB, or alternatively, will shift-in a superfluous LSB. The result may be benign (*e.g.* 2× error in amplitude) or quite drastic (*e.g.* noise due to spurious changes in sign of samples – remember, the sample MSB is a sign bit).
2. A *sawtooth wave generator* to test the DAC interface. This circuit produces a sawtooth (ramp) wave of audible frequency, represented by 31250 16-bit samples per second. You may choose your own frequency. For example, a frequency of 1000Hz can be generated by counting from 0 to 31 on a 31.25kHz clock (strictly speaking, count –16 to 15, since the DAC expects two's-complement samples). Full-amplitude 16-bit samples can be generated by placing this 5-bit count in the most significant bits and concatenating 11 trailing zeros.



**Sampled waveform at 31250 samples-per-second**

For pre-lab, create modules for the DAC interface and waveform generator. Test your design in simulation, separately and together. Verify that the DAC interface can serialize and transmit arbitrary 16-bit numbers, not just “nice round numbers” from the sawtooth waveform.

You are **strongly** advised to wire-wrap your board before your regular lab session – it is a time-consuming process. Wire-wrapping can be done based on the online wrap-ID guides even if you do not have the actual components.

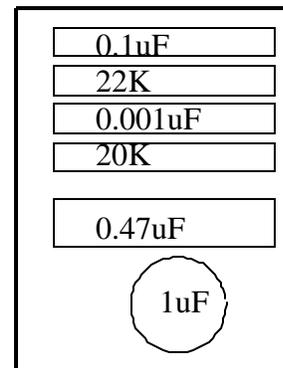
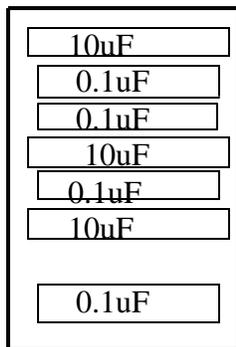
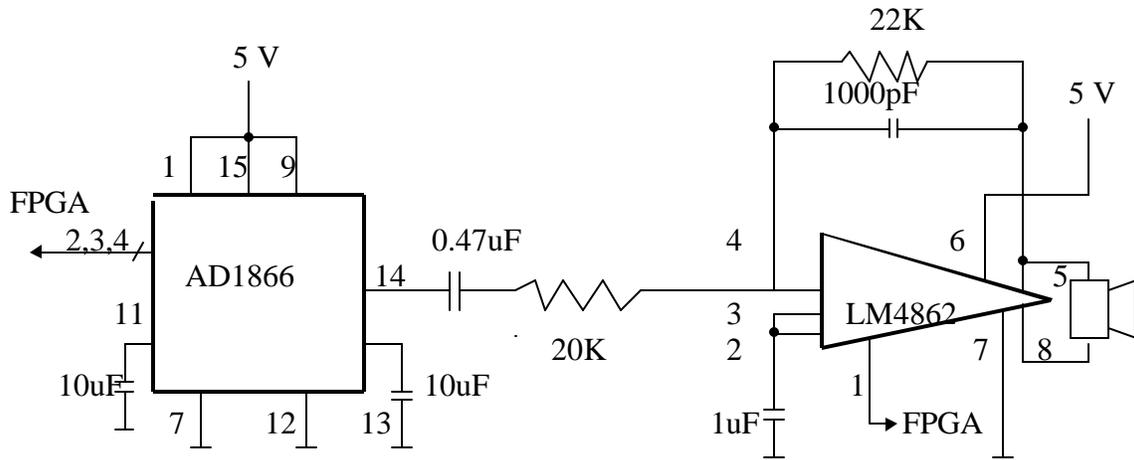
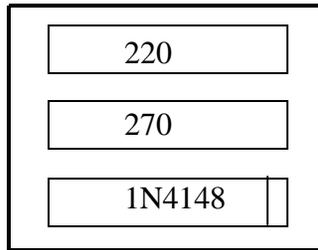
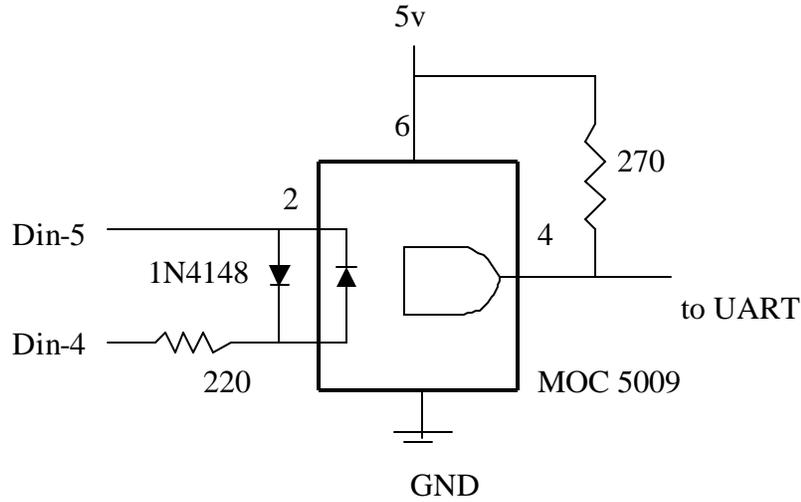
To test your audio output stage, use your waveform generator to drive the DAC interface, which will in turn drive the DAC and amplifier. View the DAC’s output (pin 14: VoL) and amplifier’s output (pin 8: Vo1 or 5: Vo2) on an oscilloscope using analog probes. Both sources should show a sawtooth wave at your target frequency, biased to oscillate around a center voltage of 2.5v. Now connect a pair of headphones to the 1/8-inch phones jack (you must provide your own headphones). You should hear a continuous buzzing tone with constant frequency and constant amplitude. Take care, the amplifier can get very loud.

## **5 Acknowledgements**

Original lab by J. Wawrzynek (Fall 1994). Rewritten by E. Caspi. Additional modifications by Jack Sampson and Yury Markovskiy.

CS150 / SPRING 2000 – Wiring of Project ICs

Tho  
Nguyen



Name: \_\_\_\_\_ Name: \_\_\_\_\_ Lab: \_\_\_\_\_

## 6 Check-offs (Checkpoint: Audio Output)

### Pre-lab

- DAC interface modules / simulation TA: \_\_\_\_\_ (20%)
- Sawtooth generator modules / simulation TA: \_\_\_\_\_ (20%)
- Combined simulation TA: \_\_\_\_\_ (10%)

### Lab Assignment

- Wire wrap DAC, amp, and discretets TA: \_\_\_\_\_ (25%)
- Working audio output (scope, headphones) TA: \_\_\_\_\_ (25%)
  
- Turned in on time TA: \_\_\_\_\_ (×100%)
- Turned in 1 week late TA: \_\_\_\_\_ (×50%)